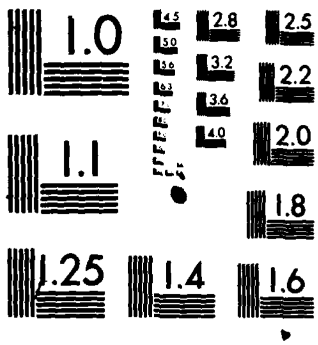
AD-A117 493 DUKE UNIV DURHAM NC DEPT OF COMPUTER SCIENCE F/G 12/1
PERFORMANCE BOUNDS FOR BINARY TESTING WITH ARBITRARY WEIGHTS.(U)
MAR 82 D W LOVELAND AFOSR-81-0221
UNCLASSIFIED CS-1982-4 AFOSR-TR-82-0557 NL

END
DATE
FILMED
8-82
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Performance Bounds for Binary Testing
With Arbitrary Weights

Donald W. Loveland

CS-1982-4

PERFORMANCE BOUNDS FOR BINARY TESTING

WITH ARBITRARY WEIGHTS

Donald W. Loveland

CS-1982-4

Current Address: Computer Science Department, Duke University, Durham, N.C., 27706

This research has been partially supported by AFOSR, Air Force Command, AFOSR 81-0221 and by the Rockland Research Center, Rockland, N.Y.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 82-0557	2. GOVT ACCESSION NO. AD-A117493	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PERFORMANCE BOUNDS FOR BINARY TESTING WITH ARBITRARY WEIGHTS		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER CS-1982-4
7. AUTHOR(s) DONALD W. LOVELAND		8. CONTRACT OR GRANT NUMBER(s) AFOSR 81-0221
9. PERFORMING ORGANIZATION NAME AND ADDRESS COMPUTER SCIENCE DEPARTMENT DUKE UNIVERSITY DURHAM NC 27706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE 61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS MATHEMATICAL & INFORMATION SCIENCES DIRECTORATE AIR FORCE OFFICE OF SCIENTIFIC RESEARCH BOLLING AFB DC 20332		12. REPORT DATE MARCH 1982
		13. NUMBER OF PAGES 38
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

SUBMITTED FOR PUBLICATION IN ACTA INFORMATICA

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

BINARY TESTING, BINARY IDENTIFICATION PROBLEM, APPROXIMATION ALGORITHMS, PERFORMANCE OF APPROXIMATION ALGORITHMS, ANALYSIS OF ALGORITHMS.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ON NEXT PAGE

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT

BINARY TESTING CONCERNS FINDING GOOD ALGORITHMS TO SOLVE THE CLASS OF BINARY IDENTIFICATION PROBLEMS. A BINARY IDENTIFICATION PROBLEM HAS AS INPUT A SET OF OBJECTS, INCLUDING ONE MARKED AS DISTINGUISHED (E.G., FAULTY), FOR EACH OBJECT AN A PRIORI ESTIMATE THAT IT IS THE DISTINGUISHED OBJECT, AND A SET OF TESTS. OUTPUT IS A TESTING PROCEDURE TO ISOLATE THE DISTINGUISHED OBJECT. ONE SEEKS MINIMAL COST TESTING PROCEDURES WHERE COST IS THE AVERAGE COST OF IOSLATION, SUMMED OVER ALL OBJECTS. THIS IS A PROBLEM SCHEMA FOR THE DIAGNOSIS PROBLEM: APPLICATIONS OCCUR IN MEDICINE, SYSTEMATIC BIOLOGY, MACHINE FAULT LOCATION, QUALITY CONTROL AND ELSEWHERE.

IN THIS PAPER WE EXTEND WORK OF GAREY AND GRAHAM TO ASSES THE CAPABILITY OF FAST APPROXIMATION RULE, THE BINARY SPLITTING RULE, TO GIVE NEAR OPTIMAL TESTING PROCEDURES WHEN THE A PRIORI ESTIMATES ARE ARBITRARY. WE FIND CONDITIONS ON THE TEST SET SUCH THAT APPROXIMATION ERROR REDUCES NEARLY TO THAT OF THE EQUALLY LIKELY A PRIORI ESTIMATE CASE OF GAREY AND GRAHAM AND FIND ANOTHER UPPER BOUND ON APPROXIMATION ERROR FOR THE SAME TEST SET CONDITION WHICH WORKS VERY WELL UNDER A PRIORI ESTIMATE ASSUMPTIONS WHERE THE FIRST RESULTS IS POOR.

2

ABSTRACT

Binary testing concerns finding good algorithms to solve the class of binary identification problems. A binary identification problem has as input a set of objects, including one marked as distinguished (e.g., faulty), for each object an *a priori* estimate that it is the distinguished object, and a set of tests. Output is a testing procedure to isolate the distinguished object. One seeks minimal cost testing procedures where cost is the average cost of isolation, summed over all objects. This is a problem schema for the diagnosis problem: applications occur in medicine, systematic biology, machine fault location, quality control and elsewhere.

In this paper we extend work of Garey and Graham to assess the capability of a fast approximation rule, the *binary splitting* rule, to give near optimal testing procedures when the *a priori* estimates are arbitrary. We find conditions on the test set such that the approximation error reduces nearly to that of the equally likely *a priori* estimate case of Garey and Graham and find another upper bound on approximation error for the same test set conditions which works very well under *a priori* estimate assumptions where the first result is poor.

Performance Bounds for Binary Testing With Arbitrary Weights

1. Introduction.

The binary testing problem is a special case of a general diagnosis problem where one seeks to find the true culprit (say, disease) from among n candidates. The general problem has been studied for many years, using Bayesian statistics, decision tables, information theory and other methods. (See Payne and Preece [9], who give a survey of the entire area.) Here we continue an investigation into the binary testing problem undertaken by Garey and Graham [5]. (Earlier work on binary testing was done by many, e.g. Chu [1], Slagle [12], Garey [2,3,4]; related work has been done by Reinwald and Soland [11], and, recently, by Moret et al. [8].)

Binary testing is the task of finding good algorithms to solve individual binary identification problems. A <u>binary identification</u> problem consists of:

a) a set O of n objects $o_1,\ldots,o_n$, some of which may be <u>distinguished</u> (e.g. faulty) objects;

b) a corresponding set of n <u>a priori</u> <u>probabilities</u> p_i (also called <u>object</u> <u>weights</u>), satisfying $p_i > 0$ where p_i is regarded as the <u>a</u> <u>priori</u> likelihood that o_i is a distinguished object;

c) a <u>test set</u> $\mathcal{T} = \{T_1,\ldots,T_m\}$ of m distinct tests over O, each test T identified with a different subset S of O such that T responds "yes" if a distinguished object is in S; otherwise T responds "no".

We henceforth assume that there is precisely one distinguished object in O. Thus we also stipulate that $\Sigma\, p_i = 1$. For any test T we write T(o) = 1 (or $o \in T$) iff the distinguished object is in the subset associated with T; otherwise T(o) = 0 (or $o \notin T$). Which object actually is the distinguished object influences neither the specification of the binary identification problem or its solution (see below), , because which object is distinguished is considered unknown and we seek a <u>procedure</u> to isolate it.

Further assumptions we make are that we have an <u>adequate</u> test set to isolate any object as distinguished, and that all tests have <u>unit cost</u> of application.

A solution to a binary identification problem is a binary decision tree that is a procedure for applying tests to determine the distinguished object. A solution is called a <u>testing procedure</u>. (The decision tree is also called a <u>solution</u> tree.) A decision tree is simply a tree graph of the possible paths to follow to isolate the distinguished object; each path is a sequence of tests and each arc from a node is chosen by the outcome of the test that labels the node. The test labeling the root is always the first test applied. Each leaf is labeled by an object name, which denotes the object isolated by the test outcomes on the path to that leaf. (We often shall replace an object name by its object weight at a leaf, whenever clarity is not impaired.)

The value of a testing procedure is its expected cost. The expected cost of a testing procedure is

$$\sum_{i=1}^{n} p_i \, l_i$$

where l_i is the path length to object o_i, i.e., the number of tests executed to isolate o_i as determined by the testing procedure.

Figure 1 presents a binary identification problem and a testing procedure with its expected cost.

In this paper we are concerned with better understanding how well a well-known algorithm for producing testing procedures does in general circumstances. The reason for study in this area is the general importance and wide applicability of the diagnosis problem, of which the binary testing problem is an important restriction. The reader is referred to an excellent survey by Payne and Preece [9] where references to applications in biology, medicine, machine fault location and pattern recognition are given, along with outlines of many approaches to finding good testing procedures. Understanding the binary testing problem took a big step forward with Garey [2],[4] where it was shown how to obtain testing procedures with minimal expected cost by use of dynamic programming algorithms. However, these dynamic programming algorithms have running time exponential in the input size (usually dominated by the listing of the tests) in the worst case. Hyafil and Rivest [6] show that the binary testing problem is NP-hard (also see Loveland [7]), which (many people believe) implies that the finding of optimal testing procedures must take exponential time in the worst case. This focuses attention on approximation algorithms for finding testing procedures, which attempt to obtain good, but not always optimal, testing procedures relatively quickly. Garey and Graham [5] studied the <u>binary splitting</u> algorithm for finding testing procedures for binary identification problems because this algorithm is the essence of several algorithms offered by earlier investigators. It is this study of the performance bounds for the binary splitting algorithm (defined below) that we extend.

We start with some needed definitions.

If $S \subseteq O$ then let $I(S) = \{i \mid o_i \in S\}$

and let

$$p(S) = \sum_{i \in I(S)} p_i.$$

Also, for test T let

$$p(T) = \sum_{i \in I(T)} p_i$$

where $I(T) = \{i \mid o_i \in T\}$.

The <u>binary splitting</u> algorithm is a rule for choosing a next test to apply at any decision point in the testing procedure. If $S \subseteq O$ and S contains the distinguished object, choose the test T_i that minimizes

$$|(p(S \cap T_i)/p(S)) - 1/2|.$$

The rationale for the binary splitting algorithm may be apparent to every computer scientist; it embodies the "divide (evenly) and conquer" approach. For the binary testing problem with unit cost tests it maximizes the reduction in uncertainty. Thus it is the restriction of various entropy-based splitting rules.

The binary splitting algorithm does not always determine a unique testing procedure because several tests may meet the selection criterion at a given point. We will consider the class of all testing procedures meeting the binary splitting algorithm condition.

The testing procedure of Figure 1 is a binary splitting testing procedure. So is the testing procedure of Figure 2, which is for the same binary identification problem and yields a better expected cost. Thus we see both that the binary splitting algorithm need not specify a unique testing procedure and that such a procedure need not be optimal.

In Section 2 certain known results are reviewed including the results of Garey and Graham which our results extend. Section 3 gives an example due to Garey and Graham that shows how bad the splitting algorithm can be for arbitrary weights. Our results follow in Sections 4 and 5.

2. Some Known Results.

It is not possible to give all the known results that relate to binary testing; a fuller summary appears in Payne and Preece [9].

A test set $\mathcal{T}$ is complete iff (if and only if) for any set $S \subseteq O$ (the set of objects) there is a test $T \in$ such that $T=S$ or $O-T=S$. If a complete test set is given, there is an algorithm (essentially the Huffman code algorithm) that determines the optimal testing procedure for arbitrary object weights. The algorithm is linear in the input string length if the input object weights are ordered so the task of quickly finding minimum expected cost testing procedures is solved in this case for the restricted problem we consider. (When tests have different costs the computation becomes much more complex, but this problem has been tackled; see Picard [10].)

The works of Garey [4], Garey and Graham [5], Hyafil and Rivest [6], etc., discussed earlier concern the incomplete test set problem. It is here that the binary splitting algorithm is used. The importance of the work of Garey and Graham [5] is that they determined a strong bound on how poorly the binary splitting testing procedures could perform relative to the optimal testing procedure when the object weights are equal. Intuition and experience with specific problems led most knowledgeable people to believe that the splitting algorithm would always produce good, if not perfect, results, especially for the equal object weight case. We shall state the results of Garey and Graham and, in the following sections, pursue the same question when the object weights are arbitrary. There the results perhaps are as surprising as the Garey and Graham results for the equal object weight problems.

Given a binary identification problem with (an incomplete) test set $\mathcal{T}$ of unit cost tests, let K_{opt} denote the expected cost of an optimal testing procedure, let K^* denote the expected cost of the lowest cost binary splitting procedure and let K' denote the expected cost of the worst (highest) cost binary splitting procedure.

The following results are proven in Garey and Graham [5]. We write log n for $\log_2 n$.

<u>Theorem</u> <u>1</u> (Garey and Graham). There exists a binary identification problem of n objects with equal object weights such that

$$\frac{K^*}{K_{opt}} > \frac{1}{10}\left[\frac{\log n}{\log \log n}\right].$$

<u>Theorem 2</u> (Garey and Graham). For any binary identification problem of n objects with equal object weights, if at most c log n tests are required to identify any object in the optimal testing procedure, then

$$\frac{K'}{K_{opt}} \leq \frac{2c \log n}{1 + \log c + \log \log n} + 2c$$

A lemma used by Garey and Graham to prove Theorem 2 is stated here also because we will have cause to refer to it. We use $|S|$ to denote the cardinality of set S.

<u>Lemma</u> (Garey and Graham). For a binary identification problem with n objects of equal object weights, if for some r, $0 < r \leq 1/2$, test set $\mathcal{T}$ satisfies the following condition:

for all $S \subseteq O$ such that $|S| \geq 2$, there exists $T \in \mathcal{T}$ such that

$r|S| \leq |S \cap T| \leq (1-r)|S|$,

then

$$K' \leq \frac{\log n}{r \log(1/r)} + \frac{1-r}{r}.$$

The special case of simply binary identification problems warrants mention because of the type of test employed. A <u>singleton</u> test responds positively to precisely one element, in set notation, $T = \{o_i\}$. We consider test sets employing singleton tests later in this paper. A <u>simple identification</u>

problem is a binary identification problem with every test a singleton test. Because the test set must be adequate an n object simple identification problem must have n-1 singleton tests, at least. For this class of binary identification problems, a fast algorithm is known for producing optimal test procedures for arbitrary cost test sets (see Garey [3]). For a related problem where at most one distinguished object exists, see Chu [1].

3. The General Case.

Garey and Graham (private communication) have discovered that the binary splitting algorithm can perform very badly relative to the optimal case under arbitrary object weights in the incomplete test set situation. The following example shows that there is a family of binary test problems such that

$$\frac{K^*}{K_{opt}} \geq \frac{n}{16}$$

for the n object member of the family. Since all reasonable testing procedures have expected cost no greater than n-1, this result is about as bad as could be expected. (A reasonable testing procedure would eliminate at least one object from consideration with each test so no path would have more than n-1 tests.)

Lower Bound on Worst Case (Garey and Graham). For pedigogical purposes it is convenient to let n be even, i.e. $n=2m+2$, $m\geq o$, and to utilize a parameter ϵ regarded as a small positive real number.

Object	Object weight
o_0	$1-\epsilon$
o_{2k-1}, $1\leq k\leq m$	$2^{-(k+1)}\epsilon$
o_{2k}, $1\leq k\leq m$	$2^{-(k+1)}\epsilon$
o_{2m+1}	$2^{-(m+1)}\epsilon$

Tests

$T_o = 0$ universe

$T_1 = \{o_{2k-1}: 1\leq k\leq m\}$

$T_2 = \{o_{2k}: 1\leq k\leq m\}$

$T_3 = \{o_1, o_2\}$

.

.

.

$T_i = \{o_{2i-5}, o_{2i-4}\}$ $3\leq i\leq m+2$

.

.

.

$T_{m+3} = \{o_{2m+1}\}$

We first find an upper bound for K_o. First, consider a path to isolate o_0. Assuming o_0 is the distinguished object we see that T_1 then T_2 then T_{m+3} eliminates all other candidates and so defines an isolating path. Every other object can be isolated within m+1 tests after T_1 or T_1 or T_2 are applied. Thus

$$(*) \quad K_{opt} \leq 3(1-\epsilon) + \epsilon(m+3)$$

$$\leq 4 \qquad \text{for } \epsilon < (m+3)^{-1} < \frac{2}{n}$$

We now consider a lower bound for K^*. Every test except T_0 has weight less than 1/2 for sufficiently small ϵ; indeed, $\bigcup_{i \neq 0} T_i < 1/2$. Thus the binary splitting algorithm will first choose the test other than T_0 with the largest weight. Test T_3 has weight $2^{-1}\epsilon$ whereas tests T_1 and T_2 have weight $(2^{-1} - 2^{-(m+2)})\,\epsilon$, as is most easily seen by observing that $T_1 \cup T_2 \cup T_{m+3}$ has weight ϵ, that T_1, T_2 and T_{m+3} are disjoint and that T_1 and T_2 are symmetrical. Let us suppose the test responds negatively. Then, in like manner, over the set $0 - \{o_1, o_2\}$, T_4 has most weight. Suppose this test result is also negative. Then we continue in this manner. Thus the binary splitting algorithm selects, in order, tests $T_3, T_4, T_5, \ldots, T_{m+2}, T_{m+3}$ in order to isolate o_0. A positive response anywhere along this path would lead to isolating other objects with path length at least one. Thus

$$(**) \quad K^* \geq m(1-\epsilon)+\epsilon$$

$$\geq \frac{n-2}{2}(1-\epsilon)+\epsilon$$

$$\geq \frac{n}{4} \text{ for } \epsilon < 1/4\,, \quad n \geq 8.$$

Putting (*) and (**) together, we have

$$\frac{K^*}{K_{opt}} \geq \frac{n}{16} \text{ for } \epsilon < \frac{2}{n}\,, \quad n \geq 8.$$

4. <u>Singleton</u> <u>Tests</u>.

In the last section we saw that for arbitrary object weights the relative performance of binary splitting procedures to optimal procedures can be about as bad as can be contemplated. In Section 2 we saw that the same ratio K'/K_{opt} for equal object weight problems was poorer than expected but not nearly so bad. In this section we find an interesting restriction of the arbitrary object weight problem set where the ratio K'/K_{opt} has nearly as good a bound as the equal weight case. The restriction is that the test set include all singleton tests.

A test set $\mathcal{T}$ is <u>singleton</u> <u>complete</u> if all singleton tests are present in $\mathcal{T}$. We shall label the test $\{o_i\}$ by T_{si}.

We recall that here all tests have unit cost and that $\log x$ denotes $\log_2 x$.

We state the first result.

<u>Theorem</u> <u>3</u>. Given a binary identification problem with n objects, $n \geq 2$, weights $p_1,\ldots,p_n$ and a singleton complete test set $\mathcal{T}$, such that some test procedure requires at most c log n tests to identify any object, then

$$\frac{K'}{K_{opt}} \leq \frac{2c \log^2 n}{1 + \log c + \log \log n} + 2c \log n$$

The upper bound here is seen to differ from the Garey and Graham result by the multiple log n. Curiously, this comes not from the bound on K' but from the lower bound on K_{opt}. A minor (but important) distinction also is that we do not want to require the <u>optimal</u> test procedure to identify any object in c log n tests but only require some test procedure to have this property. When arbitrary weights are involved a non-optimal procedure may have this property when an optimal procedure does not.

To obtain this result we use a modification of the lemma of Garey and Graham stated at the end of Section 2.

<u>Lemma</u>. Given a binary identification problem with n objects, if there exists an r, $0 < r \leq 1/2$, such that for each $S \subseteq O$ with $|S| \geq 2$ either

(a) There exists a $T_{si} \in \mathcal{T}$ such that

$$\frac{p(S \cap T_{si})}{p(S)} > 1/2$$

or

(b) There exists a $T \in \mathcal{T}$ such that

$$r \cdot p(S) \leq p(S \cap T) \leq (1-r) \cdot p(S)$$

then

$$K' \leq \frac{\log n}{r \log (1/r)} + \frac{1-r}{r}$$

<u>Proof</u> <u>of</u> <u>Lemma</u>.

The proof is by induction on n. The result is seen by inspection to hold for $n=1$, $n=2$, and $n=3$. (Note for $n=3$ that $K' < 2$.) We show that the lemma holds for each $n_o \geq 4$. To begin the induction step proof we may assume that the lemma holds for all $n < n_o$. Assume the hypotheses of the lemma hold for n_o and that the binary splitting algorithm generates a test procedure. The first test splits O into S and $\bar{S}$ of weight $p(S)$ and $p(\bar{S})$ respectively. K_S ($K_{\bar{S}}$) denotes the expected number of tests required for $S(\bar{S})$ by the algorithm.

We suppose that S and $\bar{S}$ are determined by condition (a) of the lemma and that T_{si} is the singleton test such that $S = O \cap T_{si}$ and $p(S) > 1/2$. Here $K_S = 0$ because $|S| = 1$. Therefore,

$$K \leq (1-p(S))(K_{\bar{S}}+1) + p(S) = (1-p(S))\, K_{\bar{S}} + 1.$$

By the induction hypothesis

$$K \leq (1-p(S))\ \left(\frac{\log(n-1)}{r \log\ (1/r)} + \frac{1-r}{r}\right) + 1$$

$$\leq 1/2\left(\frac{\log(n-1)}{r \log\ (1/r)} + \frac{1-r}{r}\right) + 1$$

For $1 \leq 1/3$, we note that $\frac{1-r}{r} \geq 2$, so

$$K \leq 1/2\ \left(\frac{\log\ (n-1)}{r \log\ (1/r)} + 2\right) + 1$$

$$\leq \frac{\log\ n}{r \log\ (1/r)} + \frac{1-r}{r}$$

For $1/3 \leq r \leq 1/2$, we show that the result holds for $n \geq 4$.

$$n \geq 4,$$

$$\log(n-1) \geq \log\ 3$$

$$\geq \log(1/r)\ ,\ \text{for}\ 1/3 \leq r,$$

$$\geq 2r\ \log(1/r).$$

Therefore,

$$1/2 \frac{\log(n-1)}{r \log (1/r)} \geq 1.$$

Using the induction hypothesis,

$$K \leq 1/2 \left(\frac{\log(n-1)}{r \log (1/r)} + \frac{1-r}{r}\right) + 1$$

$$\leq \frac{\log(n-1)}{r \log (1/r)} + \frac{1-r}{r}$$

Since the argument is valid for any test procedure generated by the binary splitting algorithm, the result holds with K' replacing K.

The case that condition (b) determines S and $\bar{S}$ follows exactly the argument in Garey and Graham [5]. ∎

The proof of Theorem 3 is a variant on the proof of Theorem 2 of Graham and Garey.

<u>Proof of Theorem 3</u>.

Consider a binary identification problem satisfying the theorem hypothesis. We show that the Lemma holds with

$$r = \frac{1}{2c \log n}$$

For convenience, let A = c log n.

Let S be any subset of O with $|S| \geq 2$. We show that if condition (b) of the Lemma does not hold then condition (a) must hold.

Suppose condition (b) fails for $r = \frac{1}{2A}$. That is,

$$p(S \cap T) < \frac{1}{2A} \cdot p(S) \quad , \text{ or}$$

$$p(S \cap T) > (1 - \frac{1}{2A})\ p(S) \quad , \text{ all } T \in \mathcal{T}.$$

If at most $\lfloor A \rfloor$ (the integral part of A) tests are then applied in any order, and all get appropriate responses, we can have a set S_A remaining such that

$$p(S_A) > p(S)/2.$$

S_A must be a singleton set for otherwise the hypothesis is violated that every object is identifiable within A tests by some testing procedure. But then there exists a singleton test T_{si} such that

$$p(S \cap T_{si}) > p(S)/2$$

which is condition (a) of the Lemma. Thus the Lemma is applicable under the theorem hypotheses.

Applying the Lemma with $r = 1/(2c \log n)$ we get

$$K' \le \frac{2c \log^2 n}{1+\log c+\log \log n} + 2c\log n \; - 1$$

Since we know that $K_{opt} \ge 1$ we have our result. ∎

In the proof of Theorem 3 we made a careful analysis of K' but used the immediately obvious lower bound of 1 for K_{opt}. We see by example that there is a class of binary identification problems that satisfy the conditions of Theorem 3 for which $K_{opt} < 2$ regardless of the number of objects in O. Thus the lower bound cannot be improved. (It is clear that for this class the ratio K'/K_{opt} is not close to the bound of Theorem 3. We consider this after stating the example.)

<u>Example</u>: The binary identification problems given here satisfy the hypotheses of Theorem 3 with $c = 1+\epsilon(n)$, where $\lim_{n\to\infty} \epsilon(n) = 0$, and have $K_{opt} < 2$.

Consider $O = \{o_1,\ldots,o_n\}$ with $p_i = 2^{-i}, 1 \le i \le n-1$, and $p_n = 2^{-(n-1)}$. All possible tests exist.

The potentially complete binary tree (where the minimum and maximum path lengths to leaves differ by at most one) is a possible, but non-optimal, testing procedure where every object is identifiable using $\lceil \log n \rceil$ tests. (Here $\lceil m \rceil$ = the least integer not less than m.) The expected cost here is O(log n).

In Figure 3 we present an alternate procedure with much lower (indeed optimal) expected cost. This is the testing procedure that is found by the binary splitting algorithm. The tree is long and thin (we will call it a vine; see next section) but this allows the larger weights to get closer to the root which can often result in the lowest expected cost. For this testing procedure we have

$$K < \left(\sum_{i=1}^{n-1} i\, 2^{-i} \right) + (n-1)2^{-(n-1)} < 2.$$

Thus $K_{opt} < 2$, uniformly in n.

5. Vine Testing Procedures.

The binary identification problem considered at the end of the last section points up a weakness of Theorem 3. There are binary identification problems where the optimal testing procedure has a long and "thin" decision tree which leads to a very small expected cost. The binary splitting algorithm often finds such a testing procedure which means that the bound given by Theorem 3 grossly overestimates the ratio K'/K_{opt} (if one is even fortunate enough to satisfy the "reachability" hypothesis). In this section we prove a theorem that gives another upper bound on the expected cost for the binary splitting testing procedures. This bound is particularly useful in those cases where Theorem 3 is least useful, namely, when the optimal testing procedure has a long and thin decision tree.

The theorem also leads one to conjecture that Theorem 3 is not a strong upper bound because the binary identification problems where K_{opt} < constant regardless of problem size are seen to have upper bounds on K' well below that given by Theorem 3. (We conjecture that the upper bound for K'/K_{opt} for binary identification problems with arbitrary object weights and singleton complete test sets is the same as the equal object weight case established by Garey and Graham.)

The theorem also has some intrinsic interest as a theorem on weighted binary trees.

The testing procedures we study here are vine procedures. A <u>vine</u> is a binary tree where all interior nodes lie on one branch. (We observe that for decision trees all interior nodes have two sons.) The tree of Figure 3 is a vine. For a vine each interior node is adjacent to (at least) one leaf node. A <u>vine procedure</u> is a testing procedure whose decision tree is a vine. An <u>optimal vine</u> procedure is a vine procedure such that if w_i and w_j are weights that label leaves, and $w_i > w_j$, then the w_i leaf is closer to the root than is the w_j leaf.

A binary identification problem with a singleton complete test set always has an optimal vine testing procedure, although many times the procedure may have relatively high expected cost. However, we have noted that when the object weights are quite skewed the optimal vine procedure can be of very low expected cost. It would be nice to relate the vine procedure to the binary splitting procedures, especially as follows: the optimal vine procedure expected cost (K_V) and any binary splitting procedure expected cost (K_{BS}) satisfy $K_{BS} \leq K_V$. Unfortunately, this does not always hold as Figure 4 shows us. However, this is the nature of the result we seek since this would give a good bound on binary splitting procedures when the best testing procedures have long and thin decision trees.

Although we cannot realize $K_{BS} \leq K_V$ we are able to show that matters do not get worse than is suggested by the example of Figure 4.

Theorem 4. For any binary identification problem where the test set $\mathcal{T}$ contains all singleton tests and for any binary splitting testing procedure for this problem we have

$$K_{BS} \leq K_V + 1$$

where K_{BS} is the expected cost for the binary splitting procedure and K_V is the expected cost for the optimal vine procedure.

In particular, for any binary identification problem with a singleton complete test set we have $K' \leq K_V + 1$, where K' is the worst case expected cost for binary splitting procedures.

Proof. The proof is presented entirely in terms of weighted binary trees except for one key property of BS trees we prove below. Because every testing procedure is represented by a decision tree, we may assume we are given a BS tree T_{BS} and we will show that the theorem statement holds by producing an (optimal) vine tree T_V such that $K_{BS} \leq K_V + 1$.

Before we can state the key property pertaining to BS trees we require some definitions. For any weighted binary tree (such that each interior node has two subtrees) one can choose a leaf labeled by w and consider the path of (zero or more) interior nodes between the leaf and the root of the binary tree (the w-path). Each interior node on the w-path has another subtree attached to the node, a secondary subtree of the w-path. (In Figure 4, the .49-path on the BS procedure tree has two secondary subtrees with one and seven leaves respectively.) We define the leaf weight of a weighted binary tree to be the sum of the weights of leaves of the tree.

We prove the following fact regarding BS trees; this is the only property specific to BS trees that we need.

Fact. For any BS tree T_{BS} and any weight w labeling a leaf of T_{BS}, every secondary subtree of the w-path, except possibly the subtree closest to the leaf, has leaf weight at least as large as weight w.

We prove the Fact by assuming it false and deriving a contradiction. Let α denote a node on the w-path where the secondary tree has leaf weight s, $s<w$, and α is not adjacent to

the leaf (labeled) w. Node α is the root of a tree; let the leaf weight of this tree be t. t is the weight that is "split up" at node α. We must have $w < t/2$ or else the optimal split is (w,t-w) and the node w is one subtree, violating our supposition that α is not adjacent to node w. But if $t/2 > w > s$, then (w,t-w) is closer to (1/2 , 1/2) than is (s,t-s) and would be favored by the binary splitting algorithm. But then one subtree to α again would be node w, making nodes w and α adjacent, which violates our supposition. Thus $s<w$ is impossible and the Fact is proven.

The proof of the theorem proceeds by building the given tree $\underline{T}_{BS}$ and also the corresponding $\underline{T}_V$ in stages. We define trees $\underline{T}_0, \underline{T}_{BS1}, \underline{T}_{BS2}, \ldots, \underline{T}_{BSn} = \underline{T}_{BS}$ and $\underline{T}_0, \underline{T}_{V1}, \ldots, \underline{T}_{Vn} = \underline{T}_V$ where $\underline{T}_0$ is a single node tree with weight 1, and $\underline{T}_{BS}$ and $\underline{T}_V$ are the trees of the binary splitting procedure and the optimal vine procedure respecti :ly. The proof is by induction on the number of stages.

The proof is better understood if we prove a restricted case first. We shall assume that for the given $\underline{T}_{BS}$ which we must reconstruct that there is no w-path with a secondary subtree whose leaf weight is less than w. We shall see that in this case that $K_{BSi} \leq K_{Vi}$, for all i, $1 \leq i \leq n$, and $K_{BS} \leq K_V$.

To define $\underline{T}_{BS1}$ and $\underline{T}_{V1}$ we begin with $\underline{T}_0$. Let w_1 be the largest weight in $\underline{T}_{BS}$ and find the w_1-path in $\underline{T}_{BS}$. Consider the vine defined by this w_1-path where the secondary subtrees in $\underline{T}_{BS}$

are replaced by single node subtrees with weight equal to the leaf weight of the secondary subtree it replaces. We call the single node subtrees <u>secondary nodes</u> for the w-path. (For the BS tree of Figure 4 the secondary nodes for the .49-path have weights .01 and .5 assigned respectively. See Figure 5.) The vine just defined is T_{BS1} and T_{V1}. Let $b_{11},\ldots,b_{1r_1}$ (where $r_1 \geq 1$) be the secondary nodes created in the vine defined above, enumerating from the leaf w_1. Thus here $w, b_{11},\ldots,b_{1r_1}$ are all the leaf weights (i.e., labels) for T_{BS1} and T_{V1}.

We now construct T_{BS2} and T_{V2}. To construct T_{BS2} from T_{BS1}, let w_2 be the next largest weight in T_{BS} after w_1. Find the w_2-path in T_{BS}. If $w_2 = w_1$ then $w_2 = b_{1k}$, some k, is possible and then $T_{BS2} = T_{BS1}$. Otherwise, the first portion of the w_2-path from the leaf is contained in a secondary subtree of the w_1-path of T_{BS}. Let b_{1k} label the secondary node in T_{BS1} associated with the secondary subtree containing part of the w_2-path. We form T_{BS2} from T_{BS1} by replacing node b_{1k} by the vine that completes the w_2-path in T_{BS2}. This vine has secondary nodes $b_{21},\ldots,b_{2r_2}$ to replace secondary subtrees in T_{BS} along the w_2-path where it is distinct from the w_1-path. (See Figure 5 for an illustration of T_{BS1}, T_{BS2}, and T_{V2} for the BS tree of Figure 4.)

To create T_{V2} from T_{V1} we expand b_{1k} in the identical way except we must first move (the label) b_{1k} to the node (labeled) w_1 so that the expansion of node b_{1k} to a vine results in another vine. We interchange (labels) w and b_{1k} to achieve this move. By the Fact (and our simplification assumption) $b_{1k} \geq w_1$ so the expected cost K cannot be decreased by this move, since there is a net weight change farther away from the root ("down the tree") if any change occurs. Now replace b_{1k} by the same vine replacing b_{1k} in the creation of $\underset{\sim}{T}_{BS2}$. This defines $\underset{\sim}{T}_{V2}$. We see that $K_{BS2} \leq K_{V2}$. (Recall that $K_{BS1} \leq K_{V1}$ since $\underset{\sim}{T}_{BS1} = \underset{\sim}{T}_{V1}$.)

The general outline should be clear. $\underset{\sim}{T}_{BS}$ is being "reconstructed" by expanding secondary nodes so that the subtrees are gradually rebuilt. The tree $\underset{\sim}{T}_V$ is gradually built by appending all the vines used in intermediate construction of $\underset{\sim}{T}_{BS}$ at the "end" of a previous vine, to preserve "vinehood". The general form for the restricted case can now be presented.

To construct $\underset{\sim}{T}_{BSi}$ from $\underset{\sim}{T}_{BS(i-1)}$, locate the i^{th} largest weight on $\underset{\sim}{T}_{BS}$. If w_i already labels a secondary node b_{jk} in $\underset{\sim}{T}_{BS(i-1)}$ so that the w_i-path of $\underset{\sim}{T}_{BS}$ is present in $\underset{\sim}{T}_{BS(i-1)}$ then $\underset{\sim}{T}_{BSi} = \underset{\sim}{T}_{BS(i-1)}$ and $\underset{\sim}{T}_{Vi} = \underset{\sim}{T}_{V(i-1)}$. Otherwise, the first portion of the w_i-path must replace the node b_{jk} where the w_i-path joins the portion already present in $\underset{\sim}{T}_{BS(i-1)}$. Again, secondary nodes with appropriate weights represent the secondary subtrees of the w_i-path not yet expanded. This defines $\underset{\sim}{T}_{BSi}$. To define $\underset{\sim}{T}_{Vi}$, label b_{jk} is moved to replace w_{i-1} at one of the farthest leaves of $\underset{\sim}{T}_{V(i-1)}$. However, rather than placing w_{i-1} in the b_{jk} location, we <u>bump up</u> the weights. That is, the closest weight w_a labeling a node farther from the root than (i.e. "below") b_{jk} is moved to the b_{jk} location. The w_a location now vacant is replaced by the closest weight below the w_a location. This continues and finally weight w_{i-1} moves to the vacant location left for it. Thus, we observe that if w_a and w_b label nodes, $a,b < i-1$, and $w_a > w_b$, then w_a remains closer to the root then w_b in all $\underline{T}_{Vk}$. Finally, replace b_{jk} at a "bottom" leaf by the vine added to $\underline{T}_{BS(i-1)}$ at b_{jk}; this defines $\underline{T}_{Vi}$.

We now show that $K_{BSi} \leq K_{Vi}$, assuming that $K_{BS(i-1)} \leq K_{V(i-1)}$ by induction hypothesis. Since the expected cost is a sum of components, each component the product of a weight and its distance from the root, any weight displaced an equal amount in creating $\underline{T}_{BSi}$ and $\underline{T}_{Vi}$ will have equal effect on $K_{BS(i-1)}$ and $K_{V(i-1)}$, thus preserving the inequality. Thus the replacement of b_{jk} by the same vine in the creation of $\underline{T}_{BSi}$ and $\underline{T}_{Vi}$ preserves the inequality. We must only note that moving b_{jk} to the w_{i-1} location and bumping up weights preserves the inequality. But $w_{q+1} \leq w_q$, all $q \leq i$, also $b_{jk} \geq w_j$ (by the Fact) and w_q below b_{jk} in $\underline{T}_{V(i-1)}$ implies $q>j$. It follows that "bumping up" results in no larger negative effect on the expected cost than if w_j were moved from the w_{i-1} location up to the b_{jk} location. Moving b_{jk} down to the w_{i-1} location at least offsets this negative effect so K_V is not decreased by this action. Thus $K_{BSi} \leq K_{Vi}$ holds.

We now consider the unrestricted case where the secondary subtree on a w-path closest to node w may have leaf weight less than w. If a w_i-path in T_{BS} has such a secondary subtree we shall call it the <u>w_i- runt</u> (or, simply, the <u>runt</u>), denote its leaf weight by a_i, and exclude it in our notation $b_{i1},\ldots,b_{ir_i}$ for secondary nodes of the w_i-path. This means b_{i1} denotes the leaf weight of the closest secondary subtree to w_i such that $b_{i1} \geq w_i$. The runt is too small to stand on its own; it will usually be associated with another node and we proceed as before as much as possible.

The change is easy when constructing T_{BSi} from $T_{BS(i-1)}$. If the w_i-path has a runt, then attach it to the w_i node, which then has weight $w_i + a_i = w'_i$. If no runt exists we let $a_i = 0$ for convenience. T_{BSi} then has a w'_i-path with $b_{i1},\ldots,b_{ir_i}$ as before but $b_{ij} \geq w'_i$ may not hold. If the w'_i-path requires the expansion of a w_j-runt, some $j<i$, in $T_{BS(i-1)}$ then replace the w'_j node by an (unlabeled) node with two sons, labeled w_j and a_j respectively, and then expand the a_j node as one would a b_{jk} node.

We now consider the creation of T_{Vi} from $T_{V(i-1)}$. There are three cases which we enumerate. In case (ii) we shall see that an a_j may be shifted from w'_j to b_{j1}, creating weights w_j and $b'_{j1} = b_{j1} + a_j$, respectively. Thus, in $T_{V(i-1)}$ we must also consider b'_{jk} of form $b_{jk} + a_j$. Also weights of form $w_j + a_k$ can appear, as will be seen.

The following possibilities arise in $T_{V(i-1)}$. Recall that the expansion of a node to a vine is determined by the expansion that occurs to create $\underset{\sim}{T}_{BSi}$.

Case (i). The "node" a_{i-1} needs to be expanded. Replace node w'_{i-1} by a node with sons w_{i-1} and a_{i-1} and then expand a_{j-1} in imitation of the expansion that creates $\underset{\sim}{T}_{BSi}$.

Case (ii). A node b_{jk} needs to be expanded. Since w_{i-1}, rather than w'_{i-1}, is to be "bumped up", move a_{i-1} to $b_{(i-1)1}$ to create $b'_{(i-1)1}$ as described earlier. (Note that the distance of a_{i-1} to the root is unchanged so the expected cost is unaffected by this move.) Now move b_{jk} to the w_{i-1} node and bump up the w's below b_{jk}'s old location, as before. If b'_{jk} is $b_{jk} + a_j$ then do not move a_j. After "bumping up", the óld b'_{jk} node will have weight $w_s + a_j$, for some s.

Case (iii). A "node" a_j, some $j<i-1$, needs to be expanded. The a_j is detached from its associate b_{jk} or w_s, the latter left in place, and a_j is moved down the vine. The w_{i-1} node is replaced by a node with sons $w_{i-1}$and a_j. Then a_j is expanded.

We now establish the relationship between the expected costs of K_{BSk} and K_{Vk}. The inequality that we actually show holds is

$$(*) \qquad K_{BSk} \le K_{Vk} + \sum_{h \in H_k} w_h$$

for $H_k \subseteq \{1,\ldots,n\}$, for all k, $1 \le k \le n$. It is quickly seen from our earlier argument that (*) holds for k=1 with $H_1 = 0$. We argue the induction step from i-1 to i, following the cases just enumerated. We have by induction hypothesis that (*) holds for k=i-1.

When Case (i) occurs for $\underset{\sim}{T}_{Vi}$, $\underset{\sim}{T}_{BSi}$ has been created by the same expansion applied to $\underset{\sim}{T}_{BS(i-1)}$ so the incremental change to $K_{BS(i-1)}$ and $K_{V(i-1)}$ is exactly the same. Thus (*) holds for k=i with $H_{i-1} = H_i$.

When Case (ii) occurs, the shift of a_{i-1} to $b_{(i-1)1}$ causes no consequences to the expected value and the remainder of the action is as for the restricted case considered earlier. Thus (*) holds for k=i with $H_{i-1} = H_i$.

When Case (iii) occurs, weight a_j is shifted down the vine which increases the expected value K_{Vi} without affecting K_{BSi}. This is fine. However, in $\underset{\sim}{T}_{BS(i-1)}$ a_j is part of w'_j and when split to two sons, w_i and a_j, their path length increases by one and so both weights are added (once) to $K_{BS(i-1)}$. In modifying $\underset{\sim}{T}_{V(i-1)}$, w_{i-1} is given an increased path length of one and a_j has a path length increase of at least one. But $w_j > w_{i-1}$ in general so the increase to $K_{BS(i-1)}$ can exceed the increase to $K_{V(i-1)}$ by an amount approaching w_j, if w_{i-1} is very small. To preserve the inequality we must add w_j to the right hand side of (*). With this accommodation we see that (*) holds for k=i for this case with $H_i = H_{i-1} + \{j\}$.

This concludes the cases we must consider, and (*) has been shown to hold in particular for $k=n$. But $K_{BS} = K_{BSn}$ and $K_V = K_{Vn}$. It follows that $K_{BS} \le K_V + 1$ since the sum of the weights is 1. $\blacksquare$

References

1. Chu, W.W.: A mathematical model for diagnosing system failures. IEEE Trans. Electron. Computers EC-16, 327-331 (June 1967).

2. Garey, M.R. Optimal binary decision trees for diagnostic identification problems. Ph.D. dissertation, U. of Wisconsin, Madison, June 1970.

3. __________. Simple binary identification problems. IEEE Trans. on Computers C-21, 588-590 (June 1972).

4. __________. Optimal binary identification procedures. SIAM J. Appl. Math. 23, 173-186 (1972).

5. Garey, M.R. and R.L. Graham. Performance bounds on the splitting algorithm for binary testing. Acta Informatica 3, 347-355 (1974).

6. Hyafil, L. and R.L. Rivest. Constructing optimal binary decision trees is NP-complete. Infor. Proc. Letters 5, 15-17 (1976).

7. Loveland, D.W. Selecting optimal test procedures from incomplete test sets. Proc. First Int. Symp. on Policy Anal. and Infor. Sci., Duke University, Durham, N.C., 228-235 (June 1979).

8. Moret, B.M.E., M.G. Thomason and R.C. Gonzalez. The

activity of a variable and its relation to decision trees. ACM Trans. on Progr. Lang. and Systems 2 (4), 580-595 (Oct. 1980).

9. Payne, R.W. and D.A. Preece. Identification keys and diagnostic tables: a review. Jour. of the Royal Stat. Soc. (Series A) 143 (3), 253-292 (1980).

10. Picard, C.F. Graphes et Questionaires. Tome 2 Questionaires. Gauthier-Villars, Paris (1972).

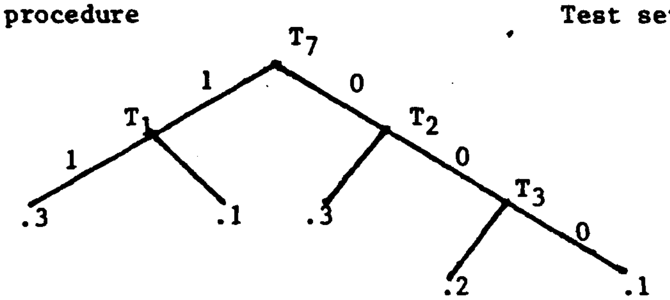
11. Reinwald, L.T. and Soland, R.M. Conversion of limited-entry decision tables to computer programs I: minimum average processing time. J.ACM 13 (3), 339-358 (July 1966).

12. Slagle, J.R. An efficient algorithm for finding certain minimum-cost procedures for making binary decisions. J.ACM 11, 253-264 (1964).

O	p_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7
o_1	.3	1	0	0	0	0	1	1
o_2	.3	0	1	0	0	0	1	0
o_3	.2	0	0	1	0	0	0	0
o_4	.1	0	0	0	1	0	0	1
o_5	.1	0	0	0	0	1	0	0

a testing procedure

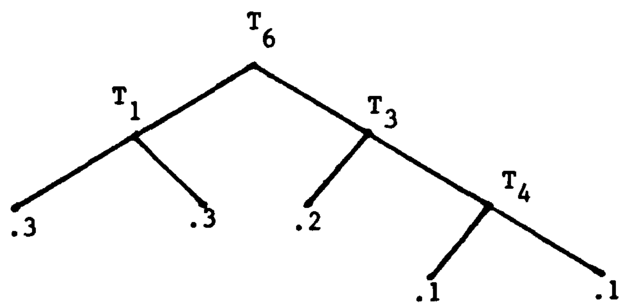
Test set = $\{T_1,\ldots,T_7\}$



$$
\begin{aligned}
\text{expected cost: } K &= 2(.3 + .1 + .3) + 3(.2 + .1) \\
&= 2(.7) + 3(.3) \\
&= 2.3
\end{aligned}
$$

A binary identification problem with one testing procedure solution

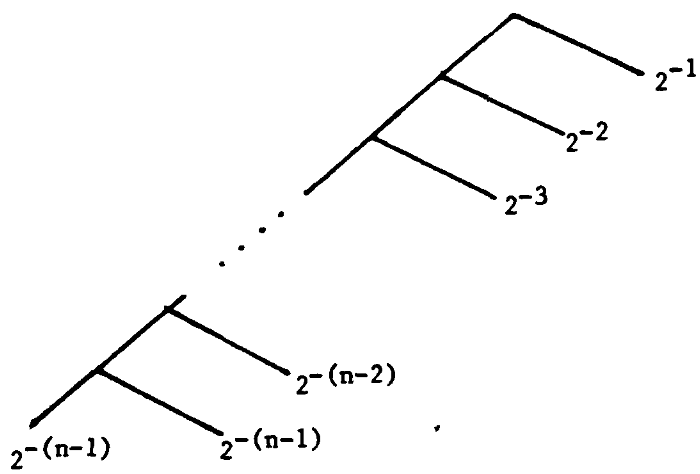
Figure 1



$$\text{expected cost:}\quad K = 2(.3 + .3 + .2) + 3(.1 + .1)$$
$$= 2(.8) + 3(.2)$$
$$= 2.2$$

An alternate testing procedure for the binary identification problem of Figure 1.

Figure 2

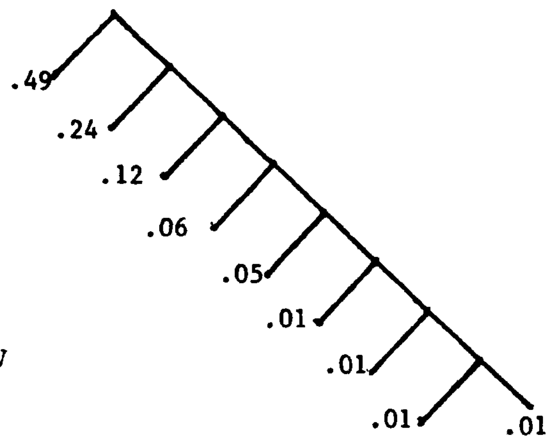


A testing procedure with expected cost less than 2.

Figure 3

Objects	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9
Weights:	.49	.24	.12	.06	.05	.01	.01	.01	.01

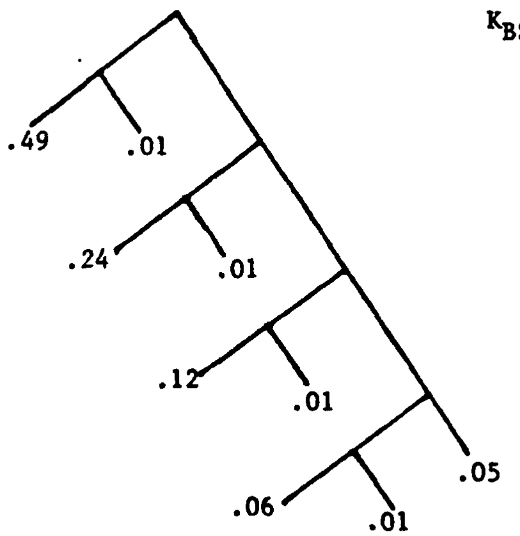
Vine procedure



$\underline{\underline{T}}_V$

$$
\begin{aligned}
K_1 &= .49 + 2(.24) + 3(.12) \\
&\quad + 4(.06) + 5(.05) \\
&\quad + (6 + 7 + 8 + 8)(.01) \\
&= 2.11
\end{aligned}
$$

Binary splitting procedure

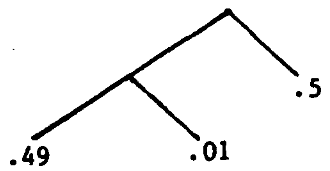


$\underline{\underline{T}}_{BS}$

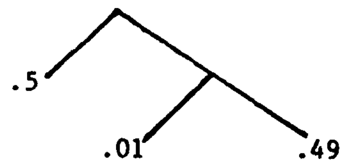
$$
\begin{aligned}
K_{BS} &= 2(.49 + .01) + \\
&\quad 3(.24 + .01) + \\
&\quad 4(.12 + .01) + \\
&\quad 4(.05) \qquad + \\
&\quad 5(.06 + .01) \\
&= 2.82
\end{aligned}
$$

A binary identification problem with nine objects.

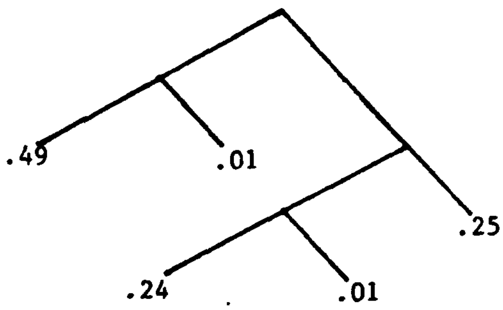
Figure 4



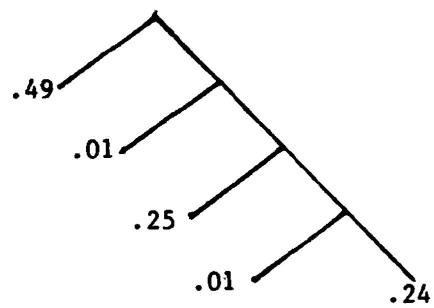
T_{BS1}



T_{V1}



T_{BS2}



T_{V2}

Construction stages associated with the tree T_{BS} of Figure 4.

Figure 5

FILMED

8 → 8